

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

KALMAN FILTERING OF FDOA/TDOA MISSILE TRACKING SYSTEM

by

Heng Cho Lin

March 2001

Thesis Advisor:

D. Curtis Schleher

Second Reader:

David C. Jenn

Approved for public release; distribution is unlimited.

20010521 164

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY		2. REPORT DATE March 2001	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: Kalman Filtering of FDOA/TDOA Missile Tracking System			5. FUNDING NUMBERS	
6. AUTHOR(S) Heng Cho Lin				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (<i>maximum 200 words</i>) <p>The accuracy of a tracking system designed to determine the time, space and position information (TSPI) of an airborne missile by detecting its telemetry signal at a number of receiver sites is investigated. Doppler frequency measurements are converted to range differences between the missile and receiver sites, whose locations are known in three dimensions. An algorithm then utilizes these range differences to obtain the missile TSPI. The accuracy of the TSPI is a function of the measurement precision and the signal-to-noise ratio at the receiver sites.</p> <p>This thesis examines the characteristics of the TSPI accuracy and investigates how a Kalman Filter can be used to enhance the accuracy of the TSPI.</p>				
14. SUBJECT TERMS Kalman Filter, Range Difference of Arrival (RDOA), Time Difference of Arrival (TDOA), Frequency Difference of Arrival (FDOA)			15. NUMBER OF PAGES 68	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

KALMAN FILTERING OF FDOA/TDOA MISSILE TRACKING SYSTEM

Heng Cho Lin

Lieutenant Colonel, Republic of Singapore Air Force
B.Eng. (Electrical Electronic Engineering), National University of Singapore, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

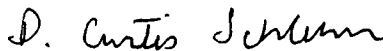
NAVAL POSTGRADUATE SCHOOL
March 2001

Author:

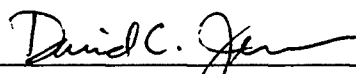


Heng Cho Lin

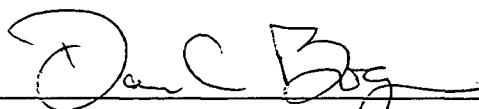
Approved by:



D. Curtis Schleher, Thesis Advisor



David C. Jenn, Second Reader



Dan C. Boger, Chairman
Computer and Information Science & Operations

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The accuracy of a tracking system designed to determine the time, space and position information (TSPI) of an airborne missile by detecting its telemetry signal at a number of receiver sites is investigated. Doppler frequency measurements are converted to range differences between the missile and receiver sites, whose locations are known in three dimensions. An algorithm then utilizes these range differences to obtain the missile TSPI. The accuracy of the TSPI is a function of the measurement precision and the signal-to-noise ratio at the receiver sites.

This thesis examines the characteristics of the TSPI accuracy and investigates how a Kalman Filter can be used to enhance the accuracy of the TSPI.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	BACKGROUND AND THEORY.....	3
A.	TELEMETRY TRACKING SYSTEM.....	3
B.	SOURCE ESTIMATION USING RANGE DIFFERENCES.....	5
C.	KALMAN FILTER.....	10
III.	MATLAB SIMULATIONS AND RESULTS.....	17
A.	TELEMETRY MISSILE TRACKING SIMULATIONS	17
B.	KALMAN FILTER SIMULATIONS	23
IV.	CONCLUSION	29
A.	SUMMARY.....	29
B.	FUTURE RESEARCH	30
	APPENDIX A. MATLAB SOURCE CODES	31
	LIST OF REFERENCES	49
	INITIAL DISTRIBUTION LIST	51

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 2.1	Notation and Geometric Relations for <i>ith</i> Receiver.	6
Figure 3.1	Missile Trajectory and Receiver Locations.	17
Figure 3.2	Missile Trajectory in the X, Y and Z Axes.	18
Figure 3.3	Telemetry Missile Tracking Simulation Flowchart.	20
Figure 3.4	Position Error vs SNR.	22
Figure 3.5	Position Error in X, Y and Z Axes.	23
Figure 3.6	Kalman Filter Simulation Flowchart.	25
Figure 3.7	Kalman Filter Simulation Output.	26
Figure 3.8	Kalman Filter on Zero Mean Noise on Z Axis Position	27
Figure 3.9	Kalman Filter on Zero Mean Noise on Z Axis Position Error.	28

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table A.1.	MATLAB Files.	31
------------	--------------------	----

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The accuracy of a tracking system designed to determine the Time, Space and Position Information (TSPI) of an airborne missile by detecting its telemetry signal at a number of remote receiver sites is investigated. The three dimensional positions of these receiver sites are known. The doppler shifted signals are received and converted to range differences to determine the telemetry source location and hence, the missile's TSPI.

This thesis studies the telemetry tracking system with 9 receiver sites that was field tested at White Sands Missile Range in 1999, evaluates the performance of the system's TSPI accuracy and examines how a Kalman filter can be used to enhance the accuracy of the TSPI. The Kalman filter is a recursive filter which minimizes the least-squares error of the system.

There are four chapters and one appendix within this thesis. In Chapter II, the theories of the telemetry tracking system and Kalman filter are introduced and discussed. Chapter III discusses the simulations of the telemetry tracking system and the Kalman filter implementation. The results of the simulations and the findings are also included in this chapter. Chapter IV summarizes and concludes this work and discusses future research options.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND THEORY

A. TELEMETRY TRACKING SYSTEM

The tracking system investigated in this thesis determines the Time, Space and Position Information (TSPI) for an airborne missile equipped with a telemetry transmitter by measuring range differences with respect to time between the missile and the receiver sites. Range differences are measured between receivers dispersed along the missile's trajectory and a reference receiver within the constellation. For N number of receivers used, there exist $N-1$ range differences that are used to compute the missile's TSPI. Range differences are computed using doppler frequencies that are obtained at each receiver site along with the telemetry signal's wavelength and sample time interval. By utilizing the doppler difference between receivers and the initial receiver location, the Range Difference Of Arrival (RDOA) is continuously determined. This RDOA is then used within an algorithm (Smith-Abel algorithm) to determine the missile's position.

The range difference determined between two receivers restricts the source location to a hyperbola with the two receivers as foci. By using 9 receiver sites, 8 independent hyperbolas are generated from 8 range differences and their intersection defines the source location.

The geometric configuration of the receivers can affect the accuracy in determining missile position. Particularly when the hyperbolas are mutually parallel, the amount of error would become very large. In the existing receivers' configuration, all the sites are close to the horizontal (X-Y) plane. If some of these sites were raised to higher heights, the system would produce better results.

Accurate doppler frequency measurement is another important factor in the system. A GPS receiver is contained at each receiver site for a time and frequency reference. During missile flight, the range to the receivers varies, causing a doppler frequency shift in the received telemetry signal at each receiver. The doppler frequency shift is realized at each receiver by the difference between the telemetry signal received from the missile and the frequency reference derived from GPS for each sample time period. Using the known wavelength of the telemetry transmitter, this doppler shift is converted to a change in range. Range tracks for each site are obtained by integrating the changes in range over time from the initial ranges obtained before launch. These individual range tracks are converted to range differences with respect to a reference receiver from which TSPI is obtained using an algorithm discussed in Chapter III.

The ever-present noise within the systems can also introduce significant errors in the estimated position. When the Signal-to-Noise Ratio (SNR) drops, the accuracy of the doppler frequency measurement is degraded. This affects the range differences and the TSPI accuracy at the output of the telemetry tracking system. As such, the TSPI accuracy is highly dependent on the SNR of the system.

Data processing is conducted upon completion of missile flight. Data from each receiver site is transmitted to the control and processing site over an RF modem data link. The modulation is removed from the data to obtain the carrier frequency with doppler shift for conversion to range tracks and then range differences to obtain the missile's TSPI.

B. SOURCE ESTIMATION USING RANGE DIFFERENCES

The algorithm for source location developed by Julius Smith and Jonathan Abel [Ref. 1] uses a spherical interpolation method with linear least-squares error minimization. It is a closed-form method that uses range differences available within a constellation of N receivers. This algorithm is used within the telemetry's post-processing system.

Let N denotes the number of receivers and let d_{ij} denotes the range difference between the i th and j th receivers and the source. The vectors of (x, y, z) spatial coordinates for the i th receiver and the source are denoted \underline{x}_i and \underline{x}_s respectively. The distance between the i th receiver and the source is defined by

$$D_i = |\underline{x}_i - \underline{x}_s| \quad (2.1)$$

and the distances from the origin to the points \underline{x}_i and \underline{x}_s , are R_i and R_s respectively.

The reference receiver is arbitrarily chosen as Receiver 1 and the coordinate system origin is translated to this receiver giving

$$\underline{x}_1 = \underline{0} \Rightarrow R_1 = 0 \quad \text{and} \quad D_1 = |\underline{x}_s| = R_s \quad (2.2)$$

Figure 2.1 illustrates the notations and geometric relations between the reference receiver (Receiver 1), the i th receiver (Receiver i), and the source. The distance between the i th receiver and the source, D_i , is found by adding the range difference between the reference receiver and the i th receiver (d_{i1}) to distance from the reference receiver to the source (R_s).

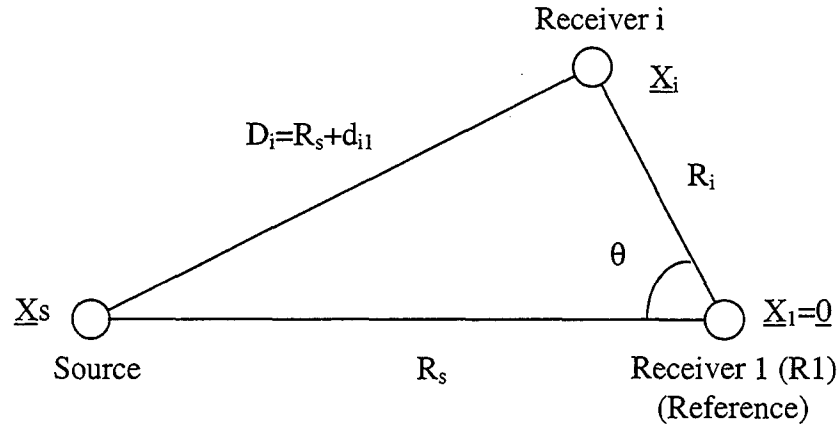


Figure 2.1. Notation and Geometric Relations for *ith* Receiver

Using the law of cosines, D_i can be expressed as

$$D_i^2 = (R_s + d_{il})^2 = R_i^2 + R_s^2 - 2\|\underline{x}_i\|\|\underline{x}_s\|\cos\theta \quad (2.3)$$

where θ is the angle between the vectors \underline{x}_i and \underline{x}_j . The angle θ between the two vectors \underline{x}_i and \underline{x}_s is defined by

$$\cos\theta = \frac{\underline{x}_i^T \underline{x}_s}{\|\underline{x}_i\|\|\underline{x}_s\|} \quad (2.4)$$

Substituting (2.4) into (2.3) and canceling out terms results in

$$(R_s + d_{il})^2 = R_i^2 + R_s^2 - 2\underline{x}_i^T \underline{x}_s \quad (2.5)$$

Expanding and shifting terms to the right-hand side of the equation yields

$$\varepsilon_i = R_i^2 - d_{il}^2 - 2R_s d_{il} - 2\underline{x}_i^T \underline{x}_s \quad (2.6)$$

where ε_i is the introduced equation error due to imprecise measurements and is minimized in a least-squares sense to provide an estimate of the true source position.

Equation (2.6) can be written in matrix form for $N-1$ range differences as

$$\underline{\varepsilon} = \underline{\delta} - 2R_s \underline{d} - 2\mathbf{S}\underline{x}_s \quad (2.7)$$

where

$$\underline{\delta} \equiv \begin{bmatrix} R_2^2 - d_{21}^2 \\ R_3^2 - d_{31}^2 \\ \vdots \\ R_N^2 - d_{N1}^2 \end{bmatrix}, \underline{d} \equiv \begin{bmatrix} d_{21} \\ d_{31} \\ \vdots \\ d_{N1} \end{bmatrix}, \mathbf{S} \equiv \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix}$$

Here \mathbf{S} represents the matrix of receiver spatial coordinates after translation of the reference receiver to the origin. The least-squares solution [Ref. 1] for \underline{x}_s given R_s is

$$\underline{x}_s = \frac{1}{2} \mathbf{S}_w^* (\underline{\delta} - 2R_s \underline{d}) \quad (2.8)$$

where * denotes complex conjugate

$$\mathbf{S}_w^* = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \quad (2.9)$$

yields the minimizer of $\underline{\varepsilon}^T \underline{\varepsilon}$. The range differences can be weighted according to the relative confidence of each range difference. Weights of zero can be used when certain range differences are not utilized. This weighted equation error is $\underline{\varepsilon}^T \mathbf{W} \underline{\varepsilon}$ [Ref. 1] and is minimized for

$$\mathbf{S}_w^* = (\mathbf{S}^T \mathbf{W} \mathbf{S})^{-1} \mathbf{S}^T \mathbf{W} \quad (2.10)$$

To obtain the source position \underline{x}_s by minimizing $\underline{\varepsilon}^T \mathbf{W} \underline{\varepsilon}$, R_s must be allowed to vary while maintaining the relation $R_s = |\underline{x}_s|$ because the range from the reference receiver to the source is not typically known in advance, resulting in non-linear minimization. This non-

linearity can be approximately eliminated by the application of the spherical interpolation method.

The goal of the spherical interpolation method is to minimize the equation error with respect to R_s . Substitution of (2.8) into (2.7) results in a linear least-squares problem for finding R_s [Ref. 1] due to the fact that the least-squares estimate of \underline{x}_s given R_s in (2.8) is linear in R_s

$$\underline{\varepsilon}' = \underline{\delta} - 2R_s \underline{d} - \mathbf{S}\mathbf{S}_w^* (\underline{\delta} - 2R_s \underline{d}) \quad (2.11)$$

where $\underline{\varepsilon}'$ is the new equation error, which is linear in R_s . Factoring out the common term $\underline{\delta} - 2R_s \underline{d}$ results in

$$\underline{\varepsilon}' = (\mathbf{I} - \mathbf{S}\mathbf{S}_w^*) (\underline{\delta} - 2R_s \underline{d}) \quad (2.12)$$

where \mathbf{I} is the identity matrix.

It follows from (2.10) that

$$\mathbf{S}\mathbf{S}_w^* = \mathbf{S}(\mathbf{S}^T \mathbf{W}\mathbf{S})^{-1} \mathbf{S}^T \mathbf{W} = \mathbf{P}_s \quad (2.13)$$

and

$$\mathbf{P}_s^\perp = \mathbf{I} - \mathbf{P}_s = \mathbf{I} - \mathbf{S}\mathbf{S}_w^* \quad (2.14)$$

where $\mathbf{S}\mathbf{S}_w^*$ defines the $N-1$ by $N-1$ symmetric matrices, and \mathbf{P}_s and \mathbf{P}_s^\perp are rank 3 projection matrices that remove both orthogonal components and components within the space spanned by the columns of \mathbf{S} , respectively.

If four receivers are utilized, three range differences are produced and $\mathbf{P}_s = \mathbf{I}$. The equation error $\underline{\varepsilon}'$ becomes zero for all values of R_s resulting in ambiguity for the source location. Because of this limitation, a minimum of five receivers producing four range differences must be used for this method to be effective in non-linear minimization.

Assuming the use of at least five receivers and four range differences, substitution of (2.14) into (2.12) results [Ref. 1] in

$$\underline{\varepsilon}' = \mathbf{P}_s^\perp (\underline{\delta} - 2R_s \underline{d}) \quad (2.15)$$

Utilizing the weighting matrix \mathbf{W} produces

$$\underline{\varepsilon}'^T \mathbf{W} \underline{\varepsilon}' = (\underline{\delta} - 2R_s \underline{d})^T \mathbf{P}_s^\perp \mathbf{W} \mathbf{P}_s^\perp (\underline{\delta} - 2R_s \underline{d}) \quad (2.16)$$

A form of weighted least squares results when minimizing (2.16) with respect to R_s , where the weighting matrix $\mathbf{P}_s^\perp \mathbf{W} \mathbf{P}_s^\perp$ is of rank $N - 4$. The three missing dimensions within this matrix result from the degrees of freedom removed by substituting (2.8) in (2.7) for the spatial coordinates of the source \underline{x}_s . The minimizing value of R_s [Ref. 1] for (2.16) is

$$\tilde{R}_s = \frac{\underline{d}^T \mathbf{P}_s^\perp \mathbf{W} \mathbf{P}_s^\perp \underline{\delta}}{2 \underline{d}^T \mathbf{P}_s^\perp \mathbf{W} \mathbf{P}_s^\perp \underline{d}} \quad (2.17)$$

Substituting (2.17) into (2.8) results [Ref. 1] in the source location estimate as

$$\underline{x}_s = \frac{1}{2} \mathbf{S}_w^* (\underline{\delta} - 2 \tilde{R}_s \underline{d}) = \frac{1}{2} (\mathbf{S}^T \mathbf{W} \mathbf{S})^{-1} \mathbf{S}^T \mathbf{W} (\underline{\delta} - 2 \tilde{R}_s \underline{d}) \quad (2.18)$$

The source's coordinates are based on a coordinate system with the reference receiver at the origin. To obtain the source's true spatial coordinates, the coordinate system must be shifted back by translation.

The Smith-Abel algorithm for source location estimation was utilized in all MATLAB simulations involving the telemetry tracking system. It was programmed into MATLAB and used as function *smith_abel.m*. The MATLAB source code for this function is listed in Appendix A.

C. KALMAN FILTER

The Kalman filter was developed in 1960. It has been applied in many diverse areas including aerospace, navigation, nuclear power plant instrumentation and many others. The Kalman filter is a recursive filter, which minimizes the least-squares error. It allows target dynamics to be used directly to optimize filter parameters. The theory of Kalman filter is fairly complex involving complicated control theories. There are many books and papers that address Kalman filter. A book written by Eli Brookner [Ref. 2] was used as the primary reference to develop the MATLAB program.

For simplicity, assume we have a one-dimensional problem. The target's velocity is constant. The target equations of motion

$$x_{n+1} = x_n + T \dot{x}_n \quad (2.19)$$

$$\dot{x}_{n+1} = \dot{x}_n \quad (2.20)$$

where x_n is the target position at time n , \dot{x}_n is the target velocity at time n , and T is the sampling time interval.

In the real world, the target will not have a constant velocity for all time. There is actually uncertainty in the target trajectory and target acceleration at any given time. The Kalman filter allows for this uncertainty in target motion by adding a random component to the target dynamics. The equations for the target dynamics are

$$x_{n+1} = x_n + T \dot{x}_n \quad (2.21)$$

$$\dot{x}_{n+1} = \dot{x}_n + u_n \quad (2.22)$$

where u_n is a random change in velocity from time n to time $n+1$.

In matrix notation, the system dynamic equation is

$$X_{n+1} = \Phi X_n + U_n \quad (2.23)$$

$$\text{where } X_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} = \text{state vector}, \quad (2.24)$$

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} = \text{state transition matrix and} \quad (2.25)$$

$$U_n = \begin{bmatrix} 0 \\ u_n \end{bmatrix} = \text{dynamic model driving noise vector.} \quad (2.26)$$

The measurement equation or the observation system equation is given as

$$Y_n = M X_n + N_n \quad (2.27)$$

$$\text{where } M = \begin{bmatrix} 1 & 0 \end{bmatrix} = \text{observation matrix}, \quad (2.28)$$

$$N_n = [V_n] = \text{observation error and} \quad (2.29)$$

$$\mathbf{Y}_n = [y_n] = \text{measurement matrix.} \quad (2.30)$$

The prediction equation is

$$\mathbf{X}_{n+1,n}^* = \Phi \mathbf{X}_{n,n}^* \quad (2.31)$$

$$\text{where } \mathbf{X}_{n,n}^* = \begin{bmatrix} x_{n,n}^* \\ \bullet \\ x_{n,n} \end{bmatrix} \text{ and} \quad (2.32)$$

$$\mathbf{X}_{n+1,n}^* = \begin{bmatrix} x_{n+1,n}^* \\ \bullet \\ x_{n+1,n} \end{bmatrix}. \quad (2.33)$$

The Kalman filtering equation [Ref. 2] is

$$\mathbf{X}_{n,n}^* = \mathbf{X}_{n,n-1}^* + \mathbf{H}_n (\mathbf{Y}_n - \mathbf{M} \mathbf{X}_{n,n-1}^*) \quad (2.34)$$

The matrix \mathbf{H}_n is known as the weight equation. It contains the filter weights for smoothing the measurement data and predicting the next position. This weight equation [Ref. 2] is

$$\mathbf{H}_n = \mathbf{S}_{n,n-1}^* \mathbf{M}^T [\mathbf{R}_n + \mathbf{M} \mathbf{S}_{n,n-1}^* \mathbf{M}^T]^{-1} \quad (2.35)$$

$$\text{where } \mathbf{S}_{n,n-1}^* = \Phi \mathbf{S}_{n-1,n-1}^* \Phi^T + \mathbf{Q}_n = \text{predictor equation} \quad (2.36)$$

$$\mathbf{Q}_n = \text{COV}[\mathbf{U}_n] = E[\mathbf{U}_n \mathbf{U}_n^T] = \text{dynamic model noise covariance} \quad (2.37)$$

$$\mathbf{S}_{n,n-1}^* = \text{COV}(\mathbf{X}_{n,n-1}^*) = E[\mathbf{X}_{n,n-1}^* \mathbf{X}_{n,n-1}^{*T}] \quad (2.38)$$

$$\mathbf{R}_n = \text{COV}(\mathbf{N}_n) = E[\mathbf{N}_n \mathbf{N}_n^T] = \text{observation noise covariance} \quad (2.39)$$

$$\mathbf{S}_{n-1,n-1}^* = \text{COV}(\mathbf{X}_{n-1,n-1}^*) = [\mathbf{I} - \mathbf{H}_{n-1} \mathbf{M}] \mathbf{S}_{n-1,n-2}^* = \text{corrector equation} \quad (2.40)$$

In the case of a maneuvering target, we need to increase the number of states in the tracking filter from 2 to 3, to include acceleration. Thus

$$X_n = \begin{bmatrix} x_n \\ \cdot \\ x_n \\ \ddots \\ x_n \end{bmatrix} \quad (2.41)$$

and the dynamics equation takes the form

$$X_{n+1} = \Phi X_n + U_n \quad (2.42)$$

The transition matrix [Ref. 2] is now given by

$$\Phi(T, \tau) = \begin{bmatrix} 1 & T & \tau^2 \left[-1 + \frac{T}{\tau} + \exp\left(-\frac{T}{\tau}\right) \right] \\ 0 & 1 & \tau \left[1 - \exp\left(-\frac{T}{\tau}\right) \right] \\ 0 & 0 & \exp\left(-\frac{T}{\tau}\right) \end{bmatrix} \quad (2.43)$$

where τ is the correlation time of the acceleration and T is the sampling interval. When T/τ is small, the target can be considered to have a constant acceleration between sample update periods. The above reduces to

$$\Phi(T, \tau) = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} = \text{Newtonian matrix} \quad (2.44)$$

The dynamic model noise covariance [Ref. 2] can be expressed as

$$Q_n = \frac{2\sigma_a^2}{\tau} \begin{bmatrix} \frac{1}{20}T^5 & \frac{1}{8}T^4 & \frac{1}{6}T^3 \\ \frac{1}{8}T^4 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \end{bmatrix} \quad (2.45)$$

The Kalman filter for this target model has an observation matrix given by

$$\mathbf{M} = [1 \quad 0 \quad 0] \quad (2.46)$$

It is initialized using

$$x_{1,1}^* = y_1, \quad \dot{x}_{1,1}^* = \frac{y_1 - y_0}{T}, \quad \ddot{x}_{1,1}^* = 0 \quad (2.47)$$

where y_0 and y_1 are the first two position measurements.

The elements of the covariance matrix for $X_{n,n}^*$ (that is $S_{n,n}^*$) are as follows:

$$[S_{1,1}^*]_{00} = \sigma_X^2 \quad (2.48)$$

$$[S_{1,1}^*]_{01} = [S_{1,1}^*]_{10} = \frac{\sigma_X^2}{T} \quad (2.49)$$

$$[S_{1,1}^*]_{02} = [S_{1,1}^*]_{20} = 0 \quad (2.50)$$

$$[S_{1,1}^*]_{11} = \frac{2\sigma_X^2}{T^2} + \frac{\sigma_a^2}{\alpha^4 T^2} \left(2 - \alpha^2 T^2 + \frac{2\alpha^3 T^3}{3} - 2e^{-\alpha T} - 2\alpha T e^{-\alpha T} \right) \quad (2.51)$$

$$[S_{1,1}^*]_{12} = [S_{1,1}^*]_{21} = \frac{\sigma_a^2}{\alpha^2 T} (e^{-\alpha T} + \alpha T - 1) \quad (2.52)$$

$$[S_{1,1}^*]_{22} = \sigma_a^2 \quad (2.53)$$

Once the initialization of the covariance matrix is completed, the Kalman filter is implemented by following 5 simple steps [Ref. 2] for each new measured position value.

Step 1. Compute the Kalman filter weighting equation to determine the filter weights.

$$\mathbf{H}_n = \mathbf{S}_{n,n-1}^* \mathbf{M}^T [\mathbf{R}_n + \mathbf{M} \mathbf{S}_{n,n-1}^* \mathbf{M}^T]^{-1} \quad (2.54)$$

Step 2. Correct the covariance matrix.

$$\mathbf{S}_{n-1,n-1}^* = \text{COV}(X_{n-1,n-1}^*) = [\mathbf{I} - \mathbf{H}_{n-1} \mathbf{M}] \mathbf{S}_{n-1,n-2}^* \quad (2.55)$$

Step 3. Update the covariance matrix.

$$\mathbf{S}_{n,n-1}^* = \Phi \mathbf{S}_{n-1,n-1}^* \Phi^T + \mathbf{Q}_n \quad (2.56)$$

Step 4. Apply the Kalman filter weights to filter the position, velocity and acceleration.

$$\mathbf{X}_{n,n}^* = \mathbf{X}_{n,n-1}^* + \mathbf{H}_n (\mathbf{Y}_n - \mathbf{M} \mathbf{X}_{n,n-1}^*) \quad (2.57)$$

Step 5. Apply the filtered position, velocity and acceleration values to predict the next position and next velocity.

$$\mathbf{X}_{n+1,n}^* = \Phi \mathbf{X}_{n,n}^* \quad (2.58)$$

THIS PAGE INTENTIONALLY LEFT BLANK

III. MATLAB SIMULATIONS AND RESULTS

A. TELEMETRY MISSILE TRACKING SIMULATIONS

The primary objective of this simulation is to examine the accuracy and characteristics of the TSPI at the telemetry tracking system output. This would enable us to implement a suitable Kalman filter to improve the accuracy of the system.

In this simulation, the actual missile trajectory data provided by a laser tracker was used to derive the different doppler frequencies at each receive sites. The missile trajectory data is as shown in Figure 3.1.

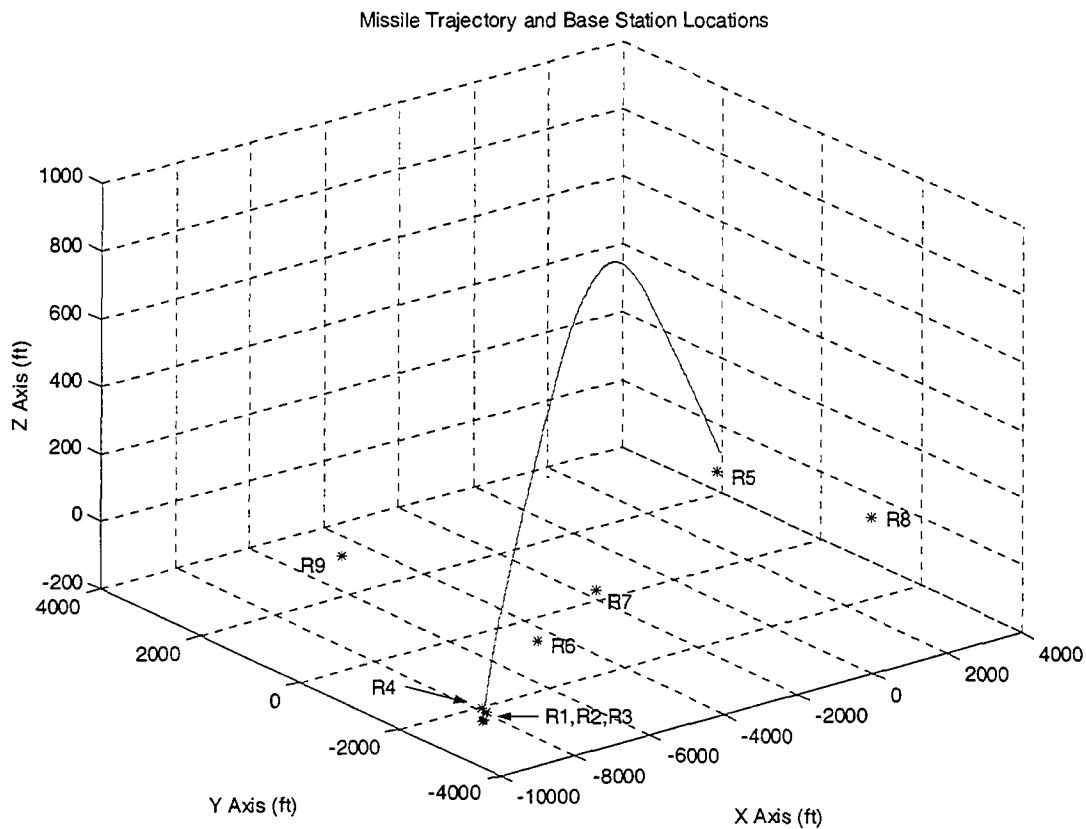


Figure 3.1. Missile Trajectory and Receiver Locations

The spatial coordinates (x , y and z) of the missile trajectory are recorded every 0.01 s. The detailed missile trajectories in the X, Y and Z axes are shown in Figure 3.2. The missile initially launches with 30 'g's acceleration force. In 6.42 sec, the missile travels 11,400 ft along the X-Axis and 3,900 ft along the Y-Axis. The missile also climbs to a maximum altitude in the Z-Axis of 1,100 ft at 3.70 sec, before descending towards the ground.

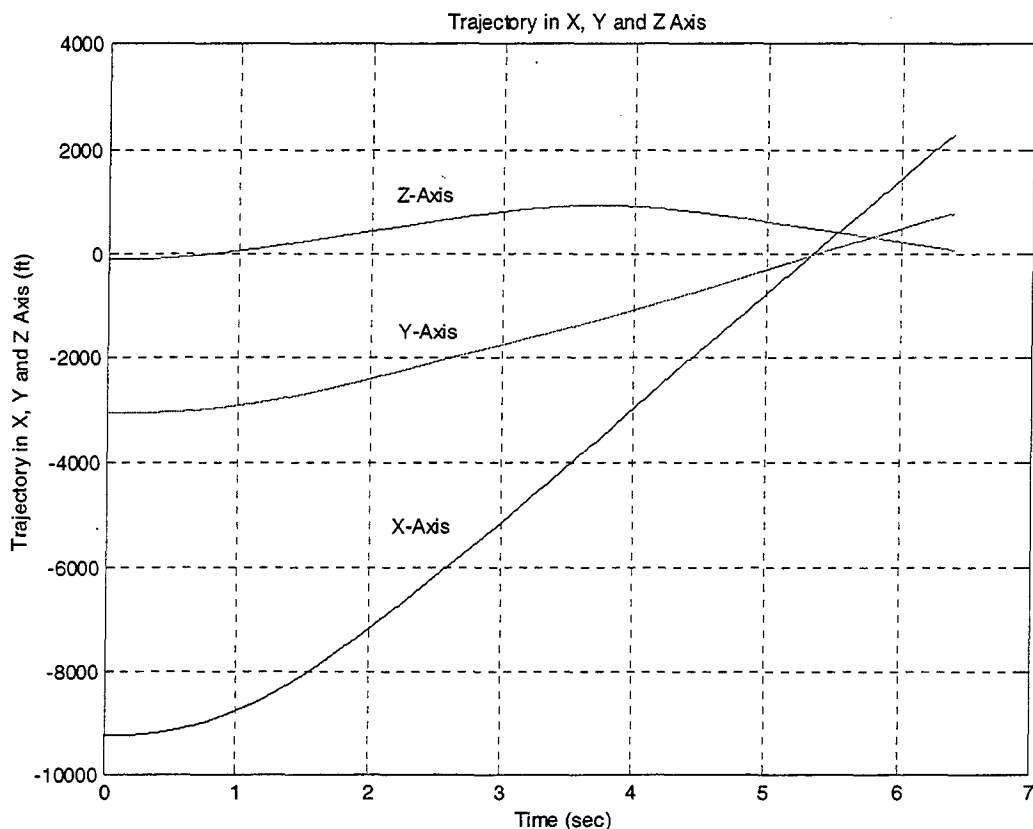


Figure 3.2. Missile Trajectory in the X, Y and Z Axes

The 9 surveyed receiver site coordinates at White Sand Missile Range were utilized for all simulations and their locations are depicted in Figure 3.1. The ranges from each receiver to the reference receiver are computed before missile flight begins because these values remain constant and are one of the inputs to the Smith-Abel

algorithm. Range Differences (RD) are found using an incremental integrating approach. The initial range differences are calculated prior to missile launch to establish a base to which average differential range differences are added at each time interval. The flowchart for the telemetry missile tracking simulation is as shown in Figure 3.3.

The phase detector frequency measurement at each of the receiver sites is degraded by the phase measurement noise and receiver noise. The theoretical lower bound on doppler frequency measurement [Ref. 3] was used, described by the following equation:

$$f = \frac{\sqrt{\frac{3}{SNR}}}{\pi \cdot T} \quad (3.1)$$

where SNR is the signal-to-noise ratio of the receiver at which the frequency measurement is made and T is the sampling period.

Each scalar doppler frequency is converted to a differential range difference. The RD between the reference and the i th receiver is defined by

$$RD_i = -\lambda \cdot T \cdot (f_{d1} - f_{di}) \quad (3.2)$$

where λ is the source wavelength, T is the sampling time, and f_{d1} and f_{di} are the respective doppler frequencies at the reference receiver and the i th receiver. The negative sign ensures a positive RD value; $(f_{d1} - f_{di})$ will always be negative because the missile is moving away from the reference receiver through its entire trajectory, producing lower relative doppler frequencies at the reference receiver.

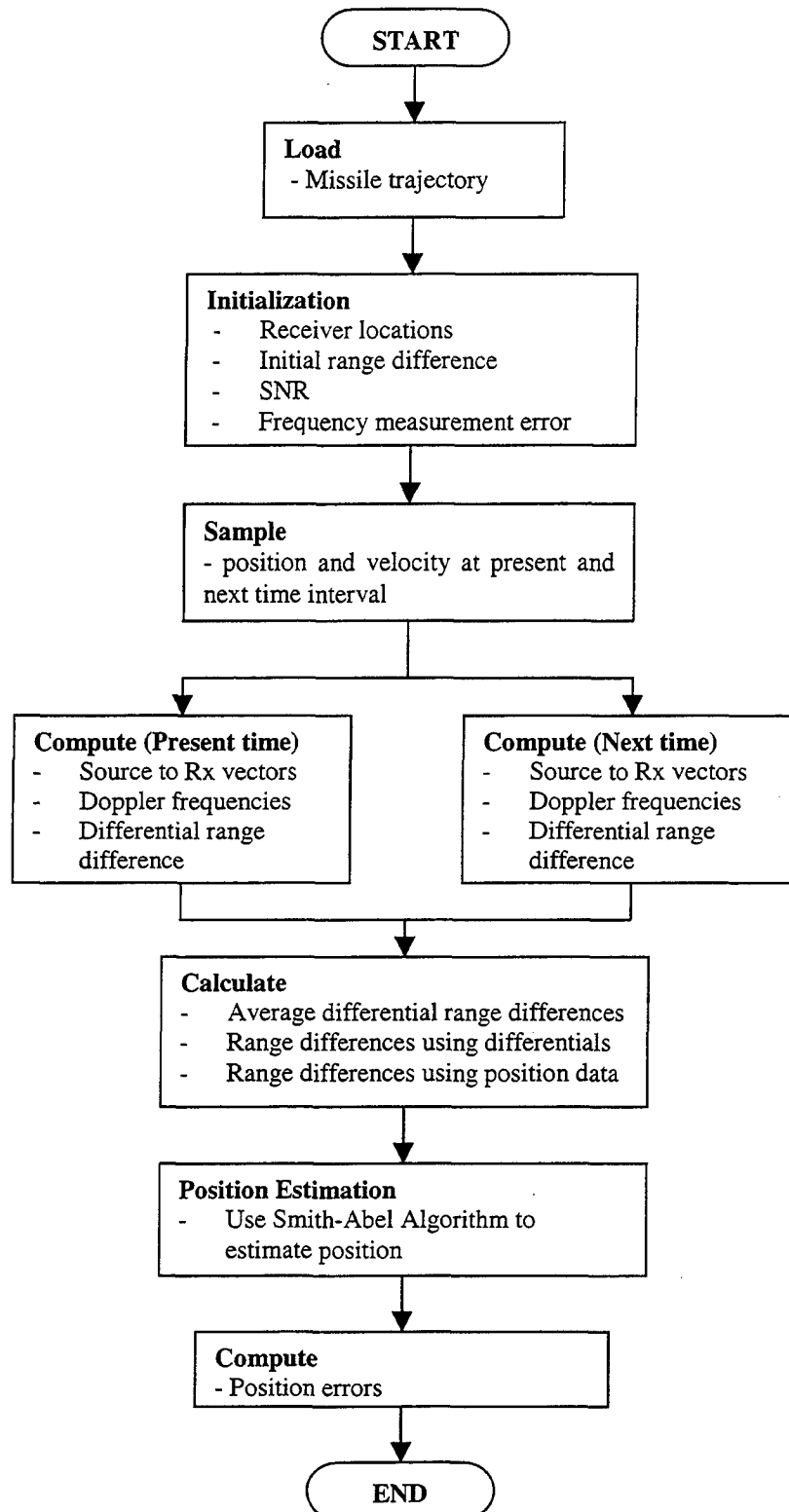


Figure 3.3. Telemetry Missile Tracking Simulation Flowchart

Two differential RDs are then computed for each receiver used. The first RD is based on missile position and velocity at the present sample time. The second RD is based on missile position and velocity at the next sample time. These sets of RDs are used to determine average differential RDs, which are added to the base RDs determined previously. The base RDs are then reset to these values and the process is repeated through the missile's trajectory. Geometric RDs are also calculated to determine the error in RD computation using missile position data for range calculations.

The missile's estimated position at the next time interval is determined using the Smith-Abel algorithm (`smith_abel.m`). Inputs to this algorithm include receiver location coordinates, RDs, and ranges from receivers to the reference receiver.

The position error is defined as the magnitude of the vector from the Smith-Abel position estimation to the missile position coordinates at the next time interval. This error is computed for each sample point along the missile's trajectory and stored. The mean value for this error is then calculated at the end of missile flight.

The mean position error is a function of the SNR. Because of the randomness of the noise, the mean position error is different for each run. We need to execute a Monte Carlo simulation of the SNR to determine an accurate mean position. The overall result of the mean position error is as shown in Figure 3.4.

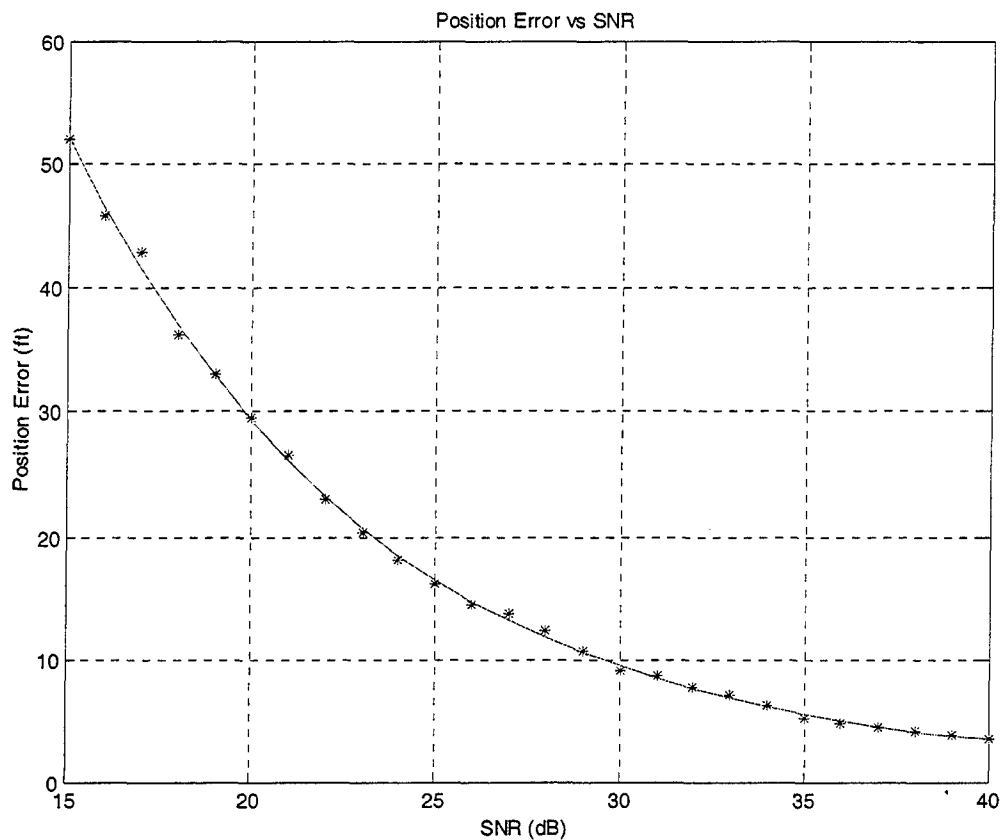


Figure 3.4. Position Error vs. SNR

In order to examine the position error in detail, the program resolves the position error into errors in the X, Y and Z Axes. A typical errors in the X, Y and Z Axes are as shown in Figure 3.5. We noted that the error in the Z-Axis is always significantly higher than those in the X-Axis and Y-Axis. The main cause of this characteristic is due to the fact that all receiver sites are close to the horizontal (X-Y) plane. This would result in higher inaccuracy in the Z-Axis that causes a larger error.

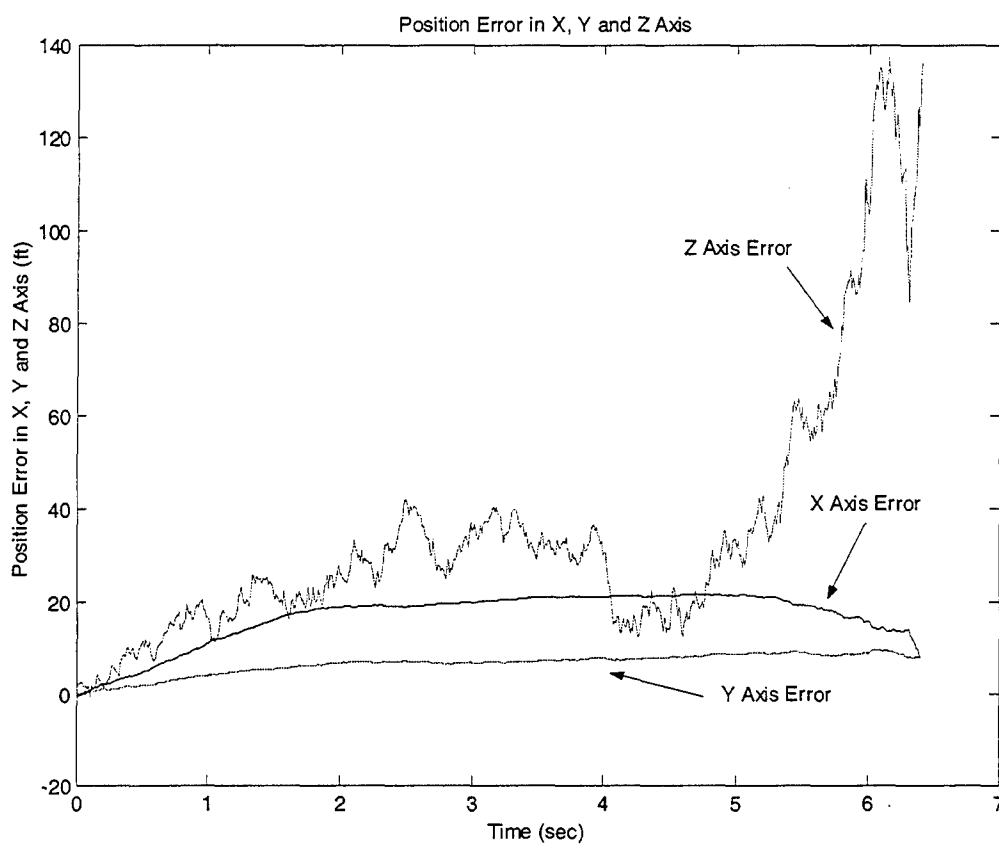


Figure 3.5. Position Error in X, Y and Z Axes

B. KALMAN FILTER SIMULATIONS

The telemetry tracking system simulation has shown that the accuracy of the TSPI is highly dependent on the SNR. The detailed examination of the position errors in the X, Y and Z Axes also revealed that the position error in the Z-Axis is the dominant component.

The approach adopted for the implementation of the Kalman filter was to process the Z-Axis position information to reduce its error. When the position error in the Z-Axis is reduced, the overall TSPI accuracy will be enhanced.

The Kalman filter simulation flowchart is as shown in Figure 3.6. The program starts off by loading the Smith-Abel estimated Z-Axis position as well as the laser tracker Z-Axis position into the program. The program would initialize the transition matrix, observation matrix, maneuver excitation covariance matrix, observations noise covariance matrix and the initial trajectory covariance matrix.

For every new estimated position input, the program will compute the Kalman filter adaptive weights. Then, the covariance matrix will be corrected and updated using equations (2.55) and (2.56).

The computed Kalman filter adaptive weights are used to filter (or smooth) the estimated position, velocity and acceleration values. Thereafter, the program will predict the next position and velocity values.

The above iteration will continue until all the estimated positions are processed. Finally, the program will compute the position errors.

In the implementation of the program, it was difficult to determine the value of the observation noise covariance. This is due to the random nature of the position error at the output of the telemetry tracking system. For a fixed value of SNR, the position error in the Z-Axis can either have a positively biased (positive mean value) or negative biased (negative mean value). A certain observation noise covariance may be good for a specific run, but it would produce undesirable results in other runs. Therefore, many 'trial and error' runs were conducted to find the optimal value for the observation noise covariance.

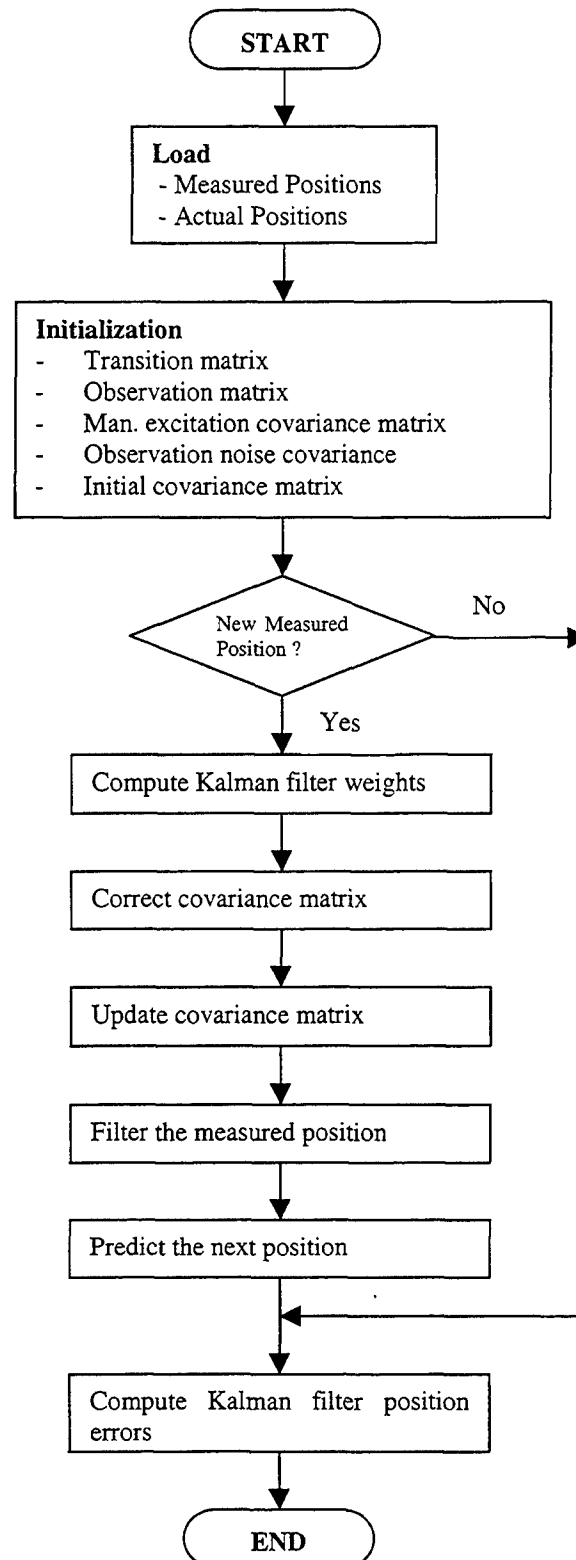


Figure 3.6. Kalman Filter Simulation Flowchart

The result of the Kalman filter on the Z-Axis position did not improved the mean position error. A typical output of the Kalman filter is as shown in Figure 3.7. The Kalman filter smooths-out the position error curve but the mean position error remains unchanged.

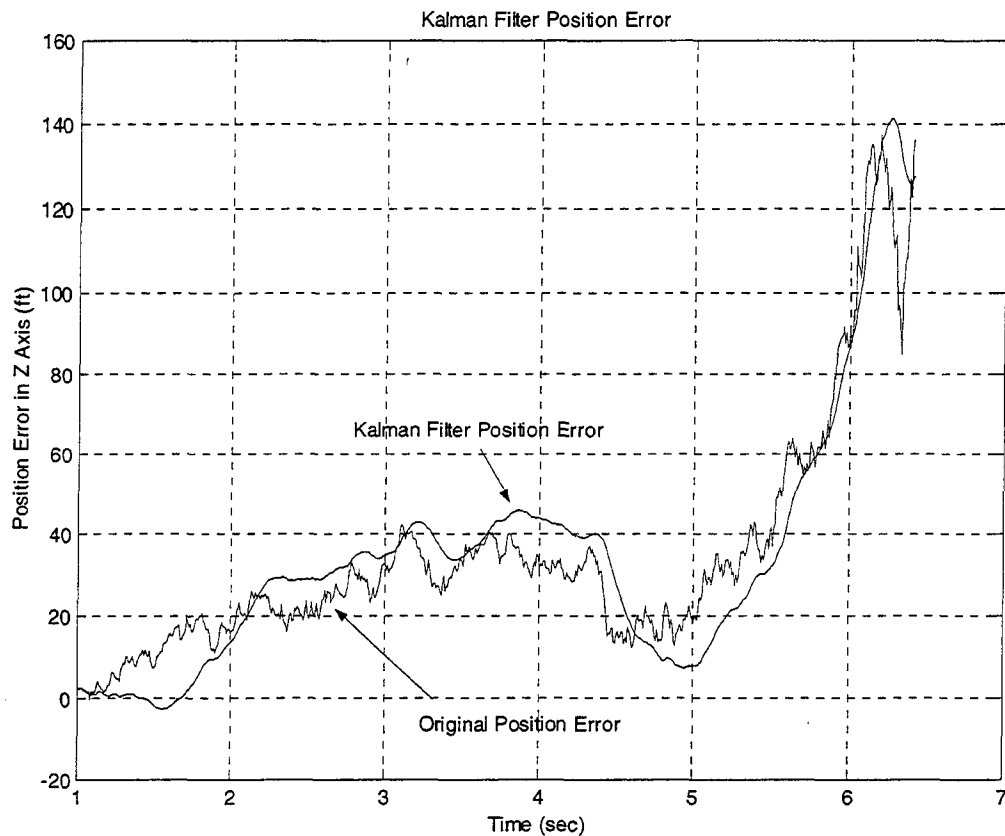


Figure 3.7. Kalman Filter Simulation Output

It was noted that the position error curve is not a zero mean Gaussian distributed variable. Depending on the randomness of the noise at the doppler frequency measurements, there may be a positively biased (positive mean value) or negative biased (negative mean value) error in the Z-Axis position curve. For example, in Figure 3.7, the Z-Axis position error changes from 0 ft at launch time to a peak value of 138 ft at 6.18

secs. This Z-Axis position error fluctuated slightly but there was a positive increasing trend. The mean position error for this run was 34.6 ft. Inputting this Z-Axis estimated position into the Kalman filter will result in a smoother fluctuation but the mean position error will not be reduced.

In order to substantiate the performance of the designed Kalman filter, a zero mean Gaussian noise was added to the laser tracker Z-Axis position for evaluation. In this case, the Kalman filter was able to smooth-out the noise fluctuation on the Z-Axis position and it effectively reduced the mean square error of the position from 79.3 ft to 19.7 ft. The Kalman filtered Z-Axis position versus the Gaussian noise imbedded position is as shown in Figure 3.8. The position errors are as shown in Figure 3.9.

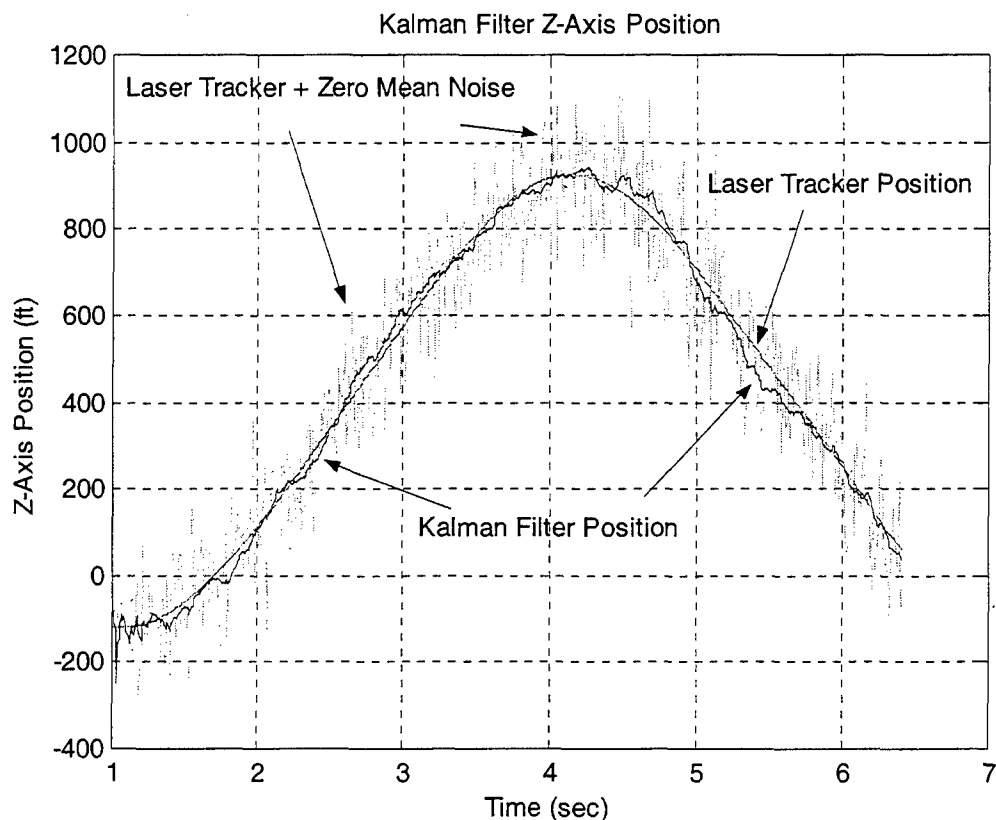


Figure 3.8. Kalman Filter on Zero Mean Noise on Z-Axis Position

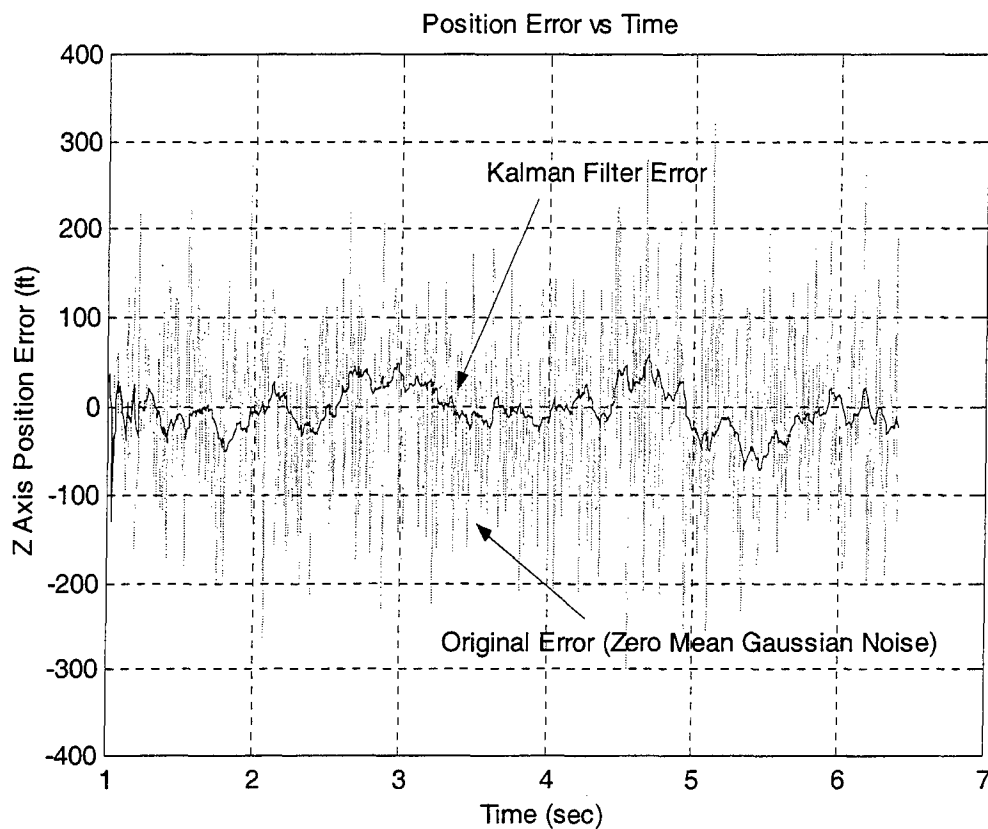


Figure 3.9. Kalman Filter on Zero Mean Noise on Z-Axis Position Error

This evaluation demonstrated the effectiveness of the Kalman filter in reducing the position error caused by zero mean noise. However, in the case of the TSPI output where the position error is not a zero mean Gaussian noise variable, the Kalman filter does not improve the position accuracy of the system.

IV. CONCLUSION

A. SUMMARY

The objective of this thesis was to examine whether a Kalman filter can be used to enhance the accuracy of the Time, Space and Position Information (TSPI) of the telemetry missile tracking system. The detailed study of the telemetry system simulation revealed that the accuracy is highly dependent on the SNR of the telemetry receivers. Closer examination of the TSPI also showed that the error in the Z-Axis is the predominant component. The approach adopted for the implementation of the Kalman filter was to process the Z-Axis position information to reduce its error. When the position error in the Z-Axis is reduced, the overall TSPI accuracy will be enhanced.

The implementation of the Kalman filter did not enhanced the accuracy of the Z-Axis position. The Kalman filter managed to smooth-out the position error curve but the mean position error remains unchanged. This is due to the fact that the position error curve is not zero mean Gaussian distributed. Depending on the randomness of the noise at the doppler frequency measurements, the Smith-Abel algorithm may produce a positively biased (positive mean value) or negative biased (negative mean value) in the Z-Axis position error curve. Additional simulation has shown that the designed Kalman filter was effective in reducing zero mean Gaussian noise.

The conclusion of this thesis is that the implementation of a Kalman filter at the TSPI output data does not enhance its accuracy.

B. FUTURE RESEARCH

We know that the phase detector frequency measurement at each of the receiver sites is degraded by the phase measurement noise and receiver noise. There is a theoretical lower bound on doppler frequency measurement and this error has zero mean Gaussian characteristics. This presents a good opportunity for the implementation of Kalman filter to improve the doppler frequency accuracy. With more accurate doppler frequency input into the telemetry tracking system, the simulation will definitely produce more accurate TSPI output.

APPENDIX A. MATLAB SOURCE CODES

The MATLAB files shown in Table A.1 were used for the simulations in support of this thesis. Their respective source codes are listed within this appendix.

No	MATLAB File	Contents
1	trajtdoasnrmc.m	Monte Carlo simulation of Smith-Abel algorithm to determine the position accuracy as a function of SNR.
2	trajtdoasnr1.m	Input missile trajectory and SNR. Compute the Smith-Abel TSPI output. Save output into data file, dd_tts1.
3	eval_tts1.m	Load data file dd_tts1 for evaluation. Resolve the error in the X, Y and Z axes. Save Z-axis position information in data file, dd_etts1.
4	kf1.m	Load data file dd_etts1 and perform Kalman filtering on the Z-axis position information.
5	zero_mean.m	Load missile trajectory and insert a Gaussian zero mean noise onto the Z-axis position information. Save noise imbedded Z-axis position information in data file, dd_zerom.
6	kf_zeromean.m	Load data file dd_zerom and perform Kalman filtering on the Z-axis position information.

Table A.1. MATLAB Files

trajtdoasnrmc.m

```
%Position from FDOA using Smith-Abel method
%Inputs missile trajectory and calculates
%RDOA from FDOA and position with S/N measurement error
%Random Gaussian noise is also present within the signal
%Monte Carlo simulation is used (50 iterations)
%trajtdoasnrmc.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vx=polyval(xvc,t);
vy=polyval(yvc,t);
vz=polyval(zvc,t);

%White Sands base station positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8 -92.1;
-8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3 -39.3; -3704.2
-1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653 -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

w=ones(8,1);
```

```

%Telemetry and sampling data
fo=2.3e9;           % Hz
wl=(3e8/fo)/.3048;   % feet
T=1e-2;             % sec
Tmax=6.41;          % sec

%Start SNR Loop for SNR values from 15-40 dB
for snrdb=15:40;
    snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);

%Start Monte Carlo loop (50 iterations)
for count=1:50;

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
    sample=ts/T+1;
    sample=round(sample);           %Sample must be an integer value
    x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
    x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
    v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
    v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
source=[x y z];

%Find source position at next time
source1=[x1 y1 z1];

%Find source-to-base station vectors at present time
B1=source-RL(1,:);
B2=source-RL(2,:);
B3=source-RL(3,:);
B4=source-RL(4,:);
B5=source-RL(5,:);
B6=source-RL(6,:);
B7=source-RL(7,:);
B8=source-RL(8,:);
B9=source-RL(9,:);

```

```

%Find Doppler frequency vector at present time
F=v/wl;

%Simulate frequency changes in telemetry at present time
fac=1+(5e5/3e9)*sign(rand-.5);

%Find Doppler frequencies at base stations at present time
fd1=fac*dot(B1,F)/norm(B1)+dfrms*randn;
fd2=fac*dot(B2,F)/norm(B2)+dfrms*randn;
fd3=fac*dot(B3,F)/norm(B3)+dfrms*randn;
fd4=fac*dot(B4,F)/norm(B4)+dfrms*randn;
fd5=fac*dot(B5,F)/norm(B5)+dfrms*randn;
fd6=fac*dot(B6,F)/norm(B6)+dfrms*randn;
fd7=fac*dot(B7,F)/norm(B7)+dfrms*randn;
fd8=fac*dot(B8,F)/norm(B8)+dfrms*randn;
fd9=fac*dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
drdoa2a=-wl*T*(fd1-fd2);
drdoa3a=-wl*T*(fd1-fd3);
drdoa4a=-wl*T*(fd1-fd4);
drdoa5a=-wl*T*(fd1-fd5);
drdoa6a=-wl*T*(fd1-fd6);
drdoa7a=-wl*T*(fd1-fd7);
drdoa8a=-wl*T*(fd1-fd8);
drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-base station vectors at next time
B1=sourcel-RL(1,:);
B2=sourcel-RL(2,:);
B3=sourcel-RL(3,:);
B4=sourcel-RL(4,:);
B5=sourcel-RL(5,:);
B6=sourcel-RL(6,:);
B7=sourcel-RL(7,:);
B8=sourcel-RL(8,:);
B9=sourcel-RL(9,:);

%Find Doppler frequency vector at next time
F1=v1/wl;

%Simulate frequency changes in telemetry at next time
fac1=1+(5e5/3e9)*sign(rand-.5);

%Find Doppler frequencies at base stations at next time
fd1=fac1*dot(B1,F1)/norm(B1)+dfrms*randn;
fd2=fac1*dot(B2,F1)/norm(B2)+dfrms*randn;
fd3=fac1*dot(B3,F1)/norm(B3)+dfrms*randn;
fd4=fac1*dot(B4,F1)/norm(B4)+dfrms*randn;
fd5=fac1*dot(B5,F1)/norm(B5)+dfrms*randn;
fd6=fac1*dot(B6,F1)/norm(B6)+dfrms*randn;
fd7=fac1*dot(B7,F1)/norm(B7)+dfrms*randn;
fd8=fac1*dot(B8,F1)/norm(B8)+dfrms*randn;
fd9=fac1*dot(B9,F1)/norm(B9)+dfrms*randn;

```

```

%Find differential RDOAs at next time
drdoa2b=-wl*T*(fd1-fd2);
drdoa3b=-wl*T*(fd1-fd3);
drdoa4b=-wl*T*(fd1-fd4);
drdoa5b=-wl*T*(fd1-fd5);
drdoa6b=-wl*T*(fd1-fd6);
drdoa7b=-wl*T*(fd1-fd7);
drdoa8b=-wl*T*(fd1-fd8);
drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
drdoa2=(drdoa2a+drdoa2b)/2;
drdoa3=(drdoa3a+drdoa3b)/2;
drdoa4=(drdoa4a+drdoa4b)/2;
drdoa5=(drdoa5a+drdoa5b)/2;
drdoa6=(drdoa6a+drdoa6b)/2;
drdoa7=(drdoa7a+drdoa7b)/2;
drdoa8=(drdoa8a+drdoa8b)/2;
drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
da(1)=d(1)+drdoa2;
da(2)=d(2)+drdoa3;
da(3)=d(3)+drdoa4;
da(4)=d(4)+drdoa5;
da(5)=d(5)+drdoa6;
da(6)=d(6)+drdoa7;
da(7)=d(7)+drdoa8;
da(8)=d(8)+drdoa9;

%Reset range differences
d=da;

%Find RDOAs from geometry
db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Estimated RDOA error
errt=norm(d - db);

%Fill matrix for estimated RDOA error
if ts==0;xet=errt;
else xet=[xet errt];
end;

%Apply Smith-Abel algorithm to find estimated position from RDOAs
source_est=smith_abel(RL,d,w,R);

%Estimated position error
errp=norm(source1 - source_est);

```



```

%Fill matrix for estimated position error
    if ts==0;xep=errp;
    else xep=[xep errp];
    end

%Return to trajectory loop
    end;

%Calculate mean TDOA and position errors for missile trajectory
%Fill matrices for these values
    merrt=mean(xet);
    merrp=mean(xep);
    if count==1;meanxet=merrt;meanxep=merrp;
    else meanxet=[meanxet merrt];meanxep=[meanxep merrp];end

%Return to Monte Carlo count loop
    end;

%Calculate and print mean values for Monte Carlo iterations
    mcet=mean(meanxet);
    mce=mean(meanxep);
    fprintf('SNR(dB) = %g\n', snrdb);
    fprintf('Error in RDOA (ft) = %g\n', mcet);
    fprintf('Error in Position (ft) = %g\n', mce);

%Fill SNR, RDOA, and position error matrices for plotting
    if snrdb==15;snrdbm=snrdb;mcetm=mcet;mce=mce;
    else snrdbm=[snrdbm snrdb];mcetm=[mcetm mcet];mce=[mce mce];end;

%Return to SNR loop
end;

```

trajtdoasnr1.m

```

%Position from FDOA using Smith-Abel method
%Inputs missile trajectory and calculates
%RDOA from FDOA and position with S/N measurement error
%Random Gaussian noise is also present within the signal
%Save data in dd_tts1 for evaluation
%trajtdoasnr1.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

```

```

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands base station positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8 -92.1;
-8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3 -39.3; -3704.2
-1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653 -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

w=ones(8,1);

%Telemetry and sampling data
fo=2.3e9; % Hz
wl=(3e8/fo)/.3048; % feet
T=1e-2; % sec
Tmax=6.41; % sec

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Input base station SNR
%snrdb=input('snr(dB)=');

```

```

snrdb=20
snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
    sample=ts/T+1;
    sample=round(sample); %Sample must be an integer value
    x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
    x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
    v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
    v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
source=[x y z];

%Find source position at next time
source1=[x1 y1 z1];

%Find source-to-base station vectors at present time
B1=source-RL(1,:);
B2=source-RL(2,:);
B3=source-RL(3,:);
B4=source-RL(4,:);
B5=source-RL(5,:);
B6=source-RL(6,:);
B7=source-RL(7,:);
B8=source-RL(8,:);
B9=source-RL(9,:);

%Find Doppler frequency vector at present time
F=v/wl;

%Simulate frequency changes in telemetry at present time
fac=1+(5e5/3e9)*sign(rand-.5);

%Find Doppler frequencies at base stations at present time
fd1=fac*dot(B1,F)/norm(B1)+dfrms*randn;
fd2=fac*dot(B2,F)/norm(B2)+dfrms*randn;
fd3=fac*dot(B3,F)/norm(B3)+dfrms*randn;
fd4=fac*dot(B4,F)/norm(B4)+dfrms*randn;
fd5=fac*dot(B5,F)/norm(B5)+dfrms*randn;
fd6=fac*dot(B6,F)/norm(B6)+dfrms*randn;
fd7=fac*dot(B7,F)/norm(B7)+dfrms*randn;
fd8=fac*dot(B8,F)/norm(B8)+dfrms*randn;
fd9=fac*dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
drdoa2a=-wl*T*(fd1-fd2);
drdoa3a=-wl*T*(fd1-fd3);
drdoa4a=-wl*T*(fd1-fd4);
drdoa5a=-wl*T*(fd1-fd5);
drdoa6a=-wl*T*(fd1-fd6);
drdoa7a=-wl*T*(fd1-fd7);

```

```

drdoa8a=-wl*T*(fd1-fd8);
drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-base station vectors at next time
B1=source1-RL(1,:);
B2=source1-RL(2,:);
B3=source1-RL(3,:);
B4=source1-RL(4,:);
B5=source1-RL(5,:);
B6=source1-RL(6,:);
B7=source1-RL(7,:);
B8=source1-RL(8,:);
B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
F1=v1/wl;

%Simulate frequency changes in telemetry at next time
fac1=1+(5e5/3e9)*sign(rand-.5);

%Find Doppler frequencies at base stations at next time
fd1=fac1*dot(B1,F1)/norm(B1)+dfirms*randn;
fd2=fac1*dot(B2,F1)/norm(B2)+dfirms*randn;
fd3=fac1*dot(B3,F1)/norm(B3)+dfirms*randn;
fd4=fac1*dot(B4,F1)/norm(B4)+dfirms*randn;
fd5=fac1*dot(B5,F1)/norm(B5)+dfirms*randn;
fd6=fac1*dot(B6,F1)/norm(B6)+dfirms*randn;
fd7=fac1*dot(B7,F1)/norm(B7)+dfirms*randn;
fd8=fac1*dot(B8,F1)/norm(B8)+dfirms*randn;
fd9=fac1*dot(B9,F1)/norm(B9)+dfirms*randn;

%Find differential RDOAs at next time
drdoa2b=-wl*T*(fd1-fd2);
drdoa3b=-wl*T*(fd1-fd3);
drdoa4b=-wl*T*(fd1-fd4);
drdoa5b=-wl*T*(fd1-fd5);
drdoa6b=-wl*T*(fd1-fd6);
drdoa7b=-wl*T*(fd1-fd7);
drdoa8b=-wl*T*(fd1-fd8);
drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
drdoa2=(drdoa2a+drdoa2b)/2;
drdoa3=(drdoa3a+drdoa3b)/2;
drdoa4=(drdoa4a+drdoa4b)/2;
drdoa5=(drdoa5a+drdoa5b)/2;
drdoa6=(drdoa6a+drdoa6b)/2;
drdoa7=(drdoa7a+drdoa7b)/2;
drdoa8=(drdoa8a+drdoa8b)/2;
drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
da(1)=d(1)+drdoa2;
da(2)=d(2)+drdoa3;
da(3)=d(3)+drdoa4;
da(4)=d(4)+drdoa5;
da(5)=d(5)+drdoa6;

```

```

da(6)=d(6)+drdoa7;
da(7)=d(7)+drdoa8;
da(8)=d(8)+drdoa9;

%Reset range differences
d=da;

%Find RDOAs from geometry
db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Fill matrices for RDOAs, time, position, and velocity components
if ts==0;dx=db;t=ts+T;xx=x;yy=y;zz=z;vx=v(1);vy=v(2);vz=v(3);
else dx=[dx db];xx=[xx x];yy=[yy y];zz=[zz z];t=[t ts+T];...
    vx=[vx v(1)];vy=[vy v(2)];vz=[vz v(3)];
end;

%Estimated RDOA error
errt=norm(d - db);

%Fill matrix for estimated RDOA error
if ts==0;xet=errt;
else xet=[xet errt];
end;

%Apply Smith-Abel algorithm to find estimated position from RDOAs
source_est=smith_abel(RL,d,w,R);

%Estimated position error
errp=norm(source1 - source_est);
x_act=source1(1,1);
y_act=source1(1,2);
z_act=source1(1,3);
x_est=source_est(1,1);
y_est=source_est(1,2);
z_est=source_est(1,3);

%Fill matrix for estimated position error
if ts==0;xep=errp;
else xep=[xep errp];
end

if ts==0;xx_est=x_est;
else xx_est=[xx_est x_est];
end

if ts==0;yy_est=y_est;
else yy_est=[yy_est y_est];
end

if ts==0;zz_est=z_est;

```

```

        else zz_est=[zz_est z_est];
    end

end;

%Obtain position and velocity components vectors
pos=[t' xx' yy' zz'];
pos=pos';
v=[t' vx' vy' vz'];
v=v';

%Output results to screen
merrrt=mean(xet);
fprintf('Mean RDOA Error (ft) = %g \n' ,merrrt);

merrrp=mean(xep);
fprintf('Mean Position Error (ft) = %g \n' ,merrrp);

%Plot graphs
figure(1)
plot(t,xet);grid;
title('RDOA Error vs Time');
xlabel('Time - sec');ylabel('RDOA error - ft');

figure(2)
plot(t,xep);grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Position Error - ft');

figure(3)
plot3(xx,yy,zz);grid;
hold on
plot3(RL(:,1),RL(:,2),RL(:,3),'*');
hold off
title('Missile Trajectory and Base Station Locations');
xlabel('ft');ylabel('ft');zlabel('ft');

%Save data in dd_tts1 for evaluation
save dd_tts1

```

eval_tts1.m

```

%load data from trajtdoasnr.m (dd_tts1) for evaluation
%output TSPI Z-axis information
%save output into dd_etts1
%eval_tts1.m

load dd_tts1

% Truth position without error
xms=xms(1,1:641);
yms=yms(1,1:641);
zms=zms(1,1:641);

% Estimated position

```

```

xx_est=xx_est(1,1:641);
yy_est=yy_est(1,1:641);
zz_est=zz_est(1,1:641);

% Position error
xerr=xx_est-xms;
yerr=yy_est-yms;
zerr=zz_est-zms;
terr=sqrt(xerr.^2+yerr.^2+zerr.^2);

mxerr=mean(xerr)
myerr=mean(yerr)
mzerr=mean(zerr)
mterr=mean(terr)

% Time scale
T=1e-2; % sec
Tmax=6.40; % sec
timel=linspace(0,Tmax,641);

figure(1)
plot(timel,zms,'k--',timel,zz_est,'k-')
title('Z Position True vs Estimated');
xlabel('Time - sec');ylabel('Z Position ft');

figure(2)
plot(timel,xms,'k--',timel,xx_est,'k-')
title('X Position True vs Estimated');
xlabel('Time - sec');ylabel('X Position ft');

figure(3)
plot(timel,yms,'k--',timel,yy_est,'k-')
title('Y Position True vs Estimated');
xlabel('Time - sec');ylabel('Y Position ft');
plot(timel,yms,timel,yy_est)

figure(4)
plot(timel,xerr,timel,yerr,timel,zerr)
title('X,Y,Z Position Error vs Time');
xlabel('Time - sec');ylabel('X,Y,Z Position ft');

%save output into dd_etts1
save dd_etts1 zms zz_est timel

```

kf1.m

```

% Execute Kalman Filtering on the Z-axis information
% load data from dd_etts1
% Kalman Filter
% output Kalman filtered Z position

format compact;clear;clf;

% load data from dd_etts1
load dd_etts1

```

```

% Re-define parameters
xm=zz_est;           % TM estimated data
xms=zms;             % Truth data
smp=length(xm);

T=1e-2;
Tl=linspace(1,smp*T,smp);

% Generate State Variables
xp=zeros(1,smp+1);
xs=zeros(1,smp);
vp=zeros(1,smp+1);
vs=zeros(1,smp);
as=zeros(1,smp);

% Initialization of variables

xs(1)=xm(1);
xp(2)=xm(1);
vs(1)=0;
vs(2)=[xm(2)-xm(1)]/T;
as(1)=0;
as(2)=0;
as(3)=[xm(3)+xm(1)-2*xm(2)]/T^2;

% Initialize Kalman Filter Covariance Matrix
sig_x2=60           % rms measurement error
sig_u2=.5           % variance of target dynamics
tau = 3e-3          % correlation time
alpha = 1/tau;

% Form Initial Covariance Matrix
S_00 = sig_x2;
S_01 = sig_x2/T;
S_11 = 2*sig_x2/T^2;
S2= [S_00 S_01 0 ; S_01 S_11 0 ; 0 0 0 ];

S=S2;

% Form Transition and Measurement Matrix

phi= [1 T T^2/2 ; 0 1 T ; 0 0 1 ];           % transition matrix
M= [1 0 0 ];                                 % observation matrix

% Form Maneuver Excitation Covariance Matrix
Q= 2*sig_u2/tau*[T^5/20 T^4/8 T^3/6 ; T^4/8 T^3/3 T^2/2 ; T^3/6
T^2/2 T ];

% Observation noise covariance
R_n = sig_x2 ;

AA=0;
BB=0;
GG=0;

for j=2:smp

```



```

% Find Kalman Filter Adaptive Weights
    D = R_n + M*S*M';      % Kalman Filter weight equation
    H_n = S*M'*inv(D);     % Kalman Filter weight equation

    S = S-H_n*M*S ;       % Correction equation

    S= phi*S*phi'+ Q ;     % Predictor covariance matrix update
    G= phi*H_n;

    Ax= G(1,1);
    Bx= G(2,1);
    Gx= G(3,1);

    AA = [AA Ax];
    BB = [BB Bx];
    GG = [GG Gx];

    % Smoothing with measured value input (Filter equation)
    xs(j)=xp(j)+Ax*(xm(j)-xp(j));
    vs(j)=vp(j)+(Bx)*(xm(j)-xp(j));
    as(j)=as(j-1)+(Gx)*(xm(j)-xp(j));

    % Predicting with smoothened value (Prediction equation)
    vp(j+1)=vs(j)+T*as(j);
    xp(j+1)=xs(j)+T*vs(j)+T^2/2*as(j);

end

% Computing the mean errors
erro = mean(abs(xm-xms))   % Original telemetry mean error
err = mean(abs(xs-xms))   % Kalman filter mean error

figure(1)
plot(Tl,xs,'b',Tl,xm,'r')
title('Smoothed Position')
xlabel('Time (sec)')
ylabel('Position (ft)')
grid,

figure(2)
plot(Tl,xm-xms,'b',Tl,xs-xms,'r');
grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Z Position Error - ft');

vs = vs(1,1:smp);
as = as(1,1:smp);

figure(3)
subplot (2,1,1)
plot(Tl,vs,'b')
xlabel('Time - sec');ylabel('Smoothed Velocity ft/s');
subplot (2,1,2)
plot(Tl,as,'b')
xlabel('Time - sec');ylabel('Smoothed Acceleration ft/s^2');

```

zero_mean.m

```
% To insert zero mean noise onto the Laser Track
% Z-axis information
% save data into dd_zerom for Kalman Filter execution
% zero_mean.m

load dd_tts1

% Truth position without error

xms=xms(1,1:641);
yms=yms(1,1:641);
zms=zms(1,1:641);

% Estimated position
xx_est=xx_est(1,1:641);
yy_est=yy_est(1,1:641);
zz_est=zz_est(1,1:641);

% Addition zero mean noise onto Z-axis position
noise=100*randn(size(zms));
zz_est=zms+noise;

% Position error
xerr=xx_est-xms;
yerr=yy_est-yms;
zerr=zz_est-zms;
terr=sqrt(xerr.^2+yerr.^2+zerr.^2);

mxerr=mean(xerr)
myerr=mean(yerr)
mzerr=mean(zerr)
mterr=mean(terr)

% Time scale
T=1e-2; % sec
Tmax=6.40; % sec
timel=linspace(0,Tmax,641);

figure(1)
plot(timel,zms,'k--',timel,zz_est,'k-')
title('Z Position True vs Estimated');
xlabel('Time - sec');ylabel('Z Position ft');

figure(2)
plot(timel,xms,'k--',timel,xx_est,'k-')
title('X Position True vs Estimated');
xlabel('Time - sec');ylabel('X Position ft');

figure(3)
plot(timel,yms,'k--',timel,yy_est,'k-')
title('Y Position True vs Estimated');
xlabel('Time - sec');ylabel('Y Position ft');
plot(timel,yms,timel,yy_est)
```

```
figure(4)
plot(time1,xerr,time1,yerr,time1,zerr)
```

```
%save output into dd_zerom
save dd_zerom zms zz_est time1
```

kf_zerom.m

```
% Execute Kalman Filter on the zero mean noise inserted
% Z-axis position
% kf_zerom.m
```

```
format compact;clear;clf;
```

```
load dd_zerom
```

```
% Re-define parameters
xm=zz_est; % TM estimated data
xms=zms; % Truth data
smp=length(xm);
```

```
T=1e-2;
Tl=linspace(1,smp*T,smp);
```

```
% Generate State Variables
```

```
xp=zeros(1,smp+1);
xs=zeros(1,smp);
vp=zeros(1,smp+1);
vs=zeros(1,smp);
as=zeros(1,smp);
```

```
% Initialization of variables
```

```
xs(1)=xm(1);
xp(2)=xm(1);
vs(1)=0;
vs(2)=[xm(2)-xm(1)]/T;
as(1)=0;
as(2)=0;
as(3)=[xm(3)+xm(1)-2*xm(2)]/T^2;
```

```
% Initialize Kalman Filter Covariance Matrix
```

```
sig_x2=60 % rms measurement error
sig_u2=.5 % variance of target dynamics
tau = 3e-3 % correlation time
alpha = 1/tau;
```

```
% Form Initial Covariance Matrix
```

```
S_00 = sig_x2;
S_01 = sig_x2/T;
S_11 = 2*sig_x2/T^2;
S2= [S_00 S_01 0 ; S_01 S_11 0 ; 0 0 0];
```

```
S=S2;
```

```

% Form Transition and Measurement Matrix

phi= [1 T T^2/2 ; 0 1 T ; 0 0 1]; % transition matrix
M= [1 0 0]; % observation matrix

% Form Maneuver Excitation Covariance Matrix
Q= 2*sig_u2/tau*[T^5/20 T^4/8 T^3/6 ; T^4/8 T^3/3 T^2/2 ; T^3/6
T^2/2 T];

% Observation noise covariance
R_n = sig_x2 ;

AA=0;
BB=0;
GG=0;

for j=2:smp

% Find Kalman Filter Adaptive Weights
D = R_n + M*S*M'; % Kalman Filter weight equation
H_n = S*M'*inv(D); % Kalman Filter weight equation

S = S-H_n*M*S ; % Correction equation

S= phi*S*phi'+ Q ; % Predictor covariance matrix update
G= phi*H_n;

Ax= G(1,1);
Bx= G(2,1);
Gx= G(3,1);

AA = [AA Ax];
BB = [BB Bx];
GG = [GG Gx];

% Smoothing with measured value input (Filter equation)
xs(j)=xp(j)+Ax*(xm(j)-xp(j));
vs(j)=vp(j)+(Bx)*(xm(j)-xp(j));
as(j)=as(j-1)+(Gx)*(xm(j)-xp(j));

% Predicting with smoothened value (Prediction equation)
vp(j+1)=vs(j)+T*as(j);
xp(j+1)=xs(j)+T*vs(j)+T^2/2*as(j);

end

% Computing the mean errors
erro = mean(abs(xm-xms)) % Original telemetry mean error
err = mean(abs(xs-xms)) % Kalman filter mean error

figure(1)
plot(Tl,xm,'g',Tl,xms,'r',Tl,xs,'k')
title('Kalman Filter Z-Axis Position')
xlabel('Time (sec)')
ylabel('Z-Axis Position (ft)')
grid,

```

```

figure(2)
plot(T1,xm-xms,'g',T1,xs-xms,'k');
grid;
title('Position Error vs Time');
xlabel('Time (sec)');ylabel('Z Axis Position Error (ft)');

vs = vs(1,1:smp);
as = as(1,1:smp);

%figure(3)
%subplot (2,1,1)
%plot(T1,vs,'b')
%xlabel('Time - sec');ylabel('Smoothed Velocity ft/s');
%subplot (2,1,2)
%plot(T1,as,'b')
%xlabel('Time - sec');ylabel('Smoothed Acceleration ft/s^2');

```

LIST OF REFERENCES

1. Abel, J.S. and Smith, J.O., "Closed Form Least Squares Source Location Estimation From Range Difference Measurements," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35,no.12, pp.1661-1669, December 1987.
2. Brookner, E., *Tracking and Kalman Filtering Made Easy*, pp.64-96, John Wiley & Sons, Inc., 1998.
3. Chestnut, P. C., "Emitter Location Accuracy Using TDOA and Differential Doppler," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-18, no. 2, pp. 215-218, March 1982.
4. Gelb, A., *Applied Optimal Estimation*, pp. 102-119, The M.I.T. Press, 1974.
5. Schleher, D.C., "Preliminary Report on Positional Accuracy of TDOA Missile system," memorandum to Naval Air Warfare Centre – Weapons Division China Lake, CA, 7 June 2000.
6. Fang, B.T., "Simple Solutions for Hyperbolic and Related Position Fixes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 5, pp. 748-753, September 1990.
7. Singer, R. A. "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-6(4), pp. 473-484, 1970.
8. Klaszky, R. "Analysis of the Positional Accuracy of a Range Difference Missile Position Measurement System," thesis report, Naval Postgraduate School, September 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, VA 22060-6218
2. Dudley Knox Library 2
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
3. Professor Curtis D. Schleher 1
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
4. Professor David C. Jenn 1
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
5. Professor Dan C. Boger 1
Chairman, Code IW
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
6. Heng Cho Lin 1
16 Jalan Arif
S548823
Singapore